



Detecting contextual hate speech code words within social media

Jherez Taylor and Prof. Yi-Shin Chen
Co-Advisor: Prof. Chaur-Chin Chen
National Tsing Hua University
Institute of Information Systems and Applications

Background

Hate Speech is “any advocacy of national, racial or religious hatred that constitutes incitement to discrimination, hostility or violence”

Why should we understand hate speech?

- Physical consequences for affected groups.

- Conditioning people to hate can be disastrous. [*1930s Germany*]

- Loss of business revenue.

Why is Hate Speech difficult to analyse?

- Hard to identify.

- Subjective perceptions of word usage.

- Frequent introduction of new terms.

Motivation

Anyone who isn't white doesn't deserve to live here. Those foreign niggers should be deported.

Use of known hate speech keyword. Easy to catch.

Motivation

Anyone who isn't white doesn't deserve to live here. Those foreign animals should be deported.

*Use of **inoffensive** keyword. Can infer the intent.*

Motivation

Anyone who isn't white doesn't deserve to live here. Those foreign skypes should be deported.

This is weird. Isn't Skype a company?

Code words act like Morse code for extremist groups.

Challenges

Code Word: *“a word or phrase that has a secret meaning or that is used instead of another word or phrase to avoid speaking directly”* Merriam-Webster

Code words are used infrequently but constantly introduced.

The context determines the hate speech meaning of a word.

Difficult to understand using a fixed dictionary, dynamic vocabulary.

Objective

Dynamically **identify** out-of-dictionary hate speech **code words**

Consider word usage based on different corpora.

Use hate speech word neighbours to expand the dictionary.

Through the context of word **similarity** and **relatedness**.

Related Work

Stereotypes and Othering Language (“Us vs Them”)

“Detecting hate speech on the world wide web” [Warner and Hirschberg 2012]
“Us and them: identifying cyber hate on Twitter” [Burnap and Williams 2016]

Need for annotation of paragraphs stereotypes.
Us vs Them terms are mostly event specific.

Code word classification

“Detecting the Hate Code on Social Media” [R. Magu et al., 2017]
Used a static list of code words, unable to identify new ones.

Neural Embedding models

“Hate Speech Detection with Comment Embeddings” [Djuric, N. et al., 2015]
“Abusive Language Detection in Online User Content” [C. Nobata. et al, 2016]
Not fit for low frequency words.



Methodology

The importance of data partitioning

There are communities of users that share hateful content

Primary source for new code words.

We assume code words here have a higher document frequency.

HateWords = $\{hw_1, hw_2, \dots, hw_n\}$ where hw_n refers to a known hate speech word

Obtained from the HateBase Organisation

Create partitions

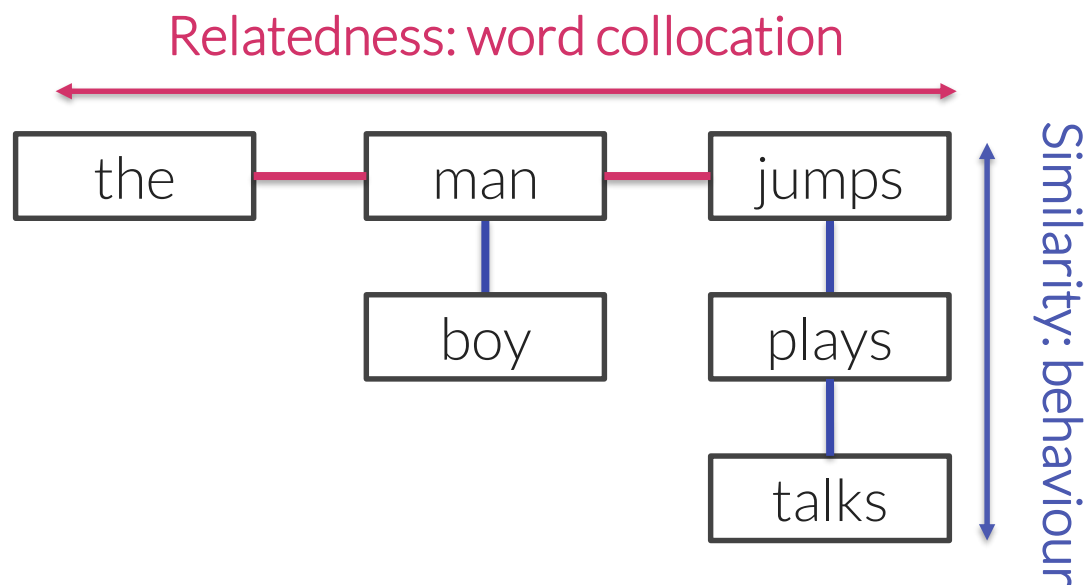
HateCommunity: articles and the tweets of users in the authour network.

CleanTexts: tweets collected without terms, filtering against **HateWords**.

Partition the data to reflect word usage under contexts.

What is word context?

Distributional hypothesis: words that appear in the same contexts share semantic meaning [Z.S. Harris, 1954]



Context modeling can allow us to find neighbouring hate speech words.

Learning Context with Neural Embeddings

Map words to vector space.

Learn relatedness with *word2vec*

Predict words that often appear together with a target word **w**.

Context: **w** and the surrounding words in a given window.

[Mikolov et al. 2016]

Learn similarity with a variation of *word2vec* called *dependency2vec*

Context: syntactic dependencies in a sentence.

SD describe the structure of a sentence.

[Levy and Goldberg ACL '14]

Why does word context matter?

| | skypes | |
|---------------|--|---|
| CleanTexts | skyped facetime Skype-ing phone | whatsapp line snapchat imessage |
| HateCommunity | chat dropbox kike Line | cockroaches negroes facebook animals |
| | Relatedness | Similarity |

The presence of **hate speech** keywords under skypes is surprising.

Create different embeddings to capture different word usage.

Build similarity and relatedness embeddings for **HateCommunity** and **CleanTexts**.

Intuition

Use word frequency in our datasets as well as context to find code words.

high frequency in hate speech data + hate speech neighbours = possible code word

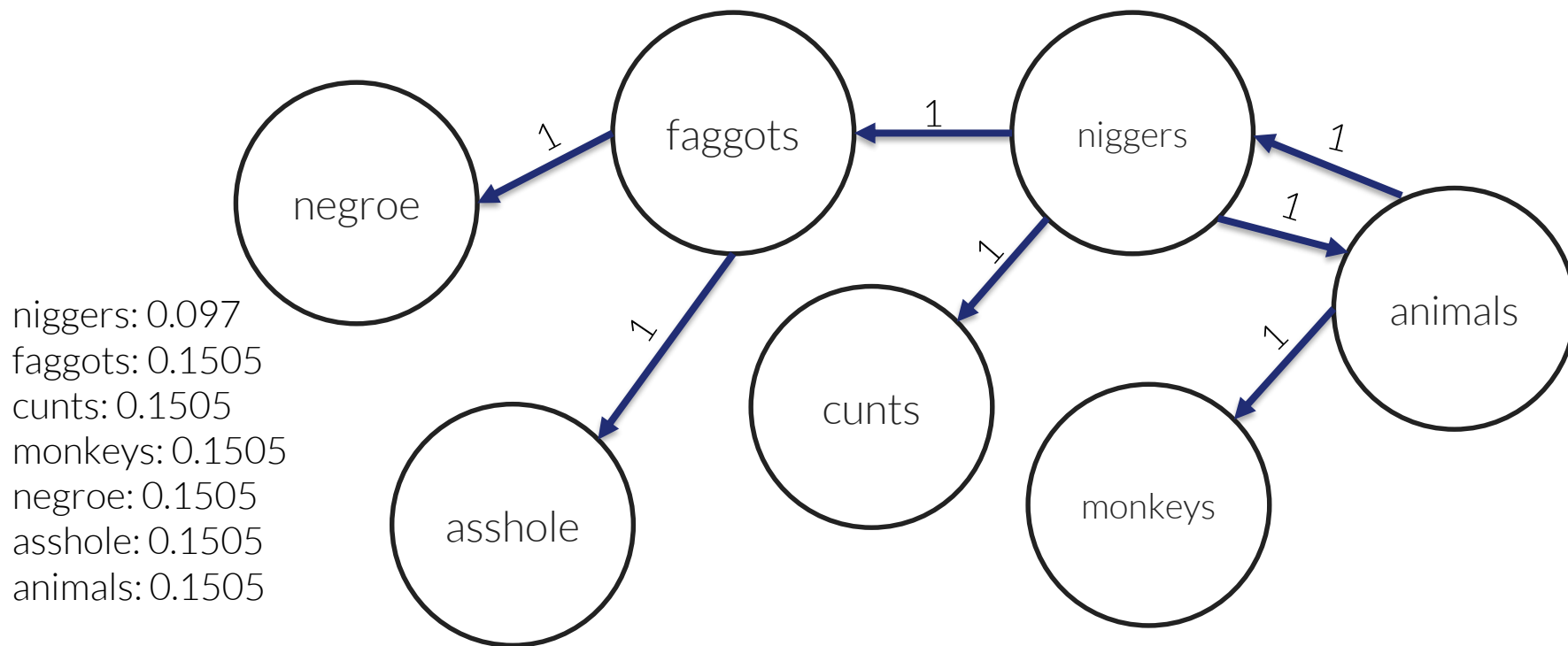
Generate an initial list of words similar to HateWords.

Impractical to do an exhaustive search of the vocabulary.

Need a way to generate neighbouring words as input.

We can use PageRank.

Finding word neighbours with PageRank



Repeat process
for leaf vertices.

Directed graph based on
word similarity from
HateCommunity

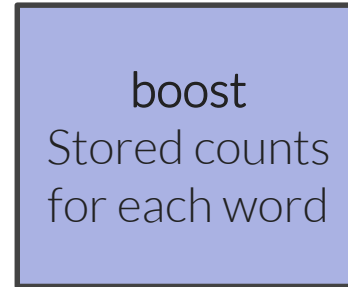
All edges have the same weight.
If we run PageRank all vertices would
have the same value.

HateWords are seed vertices

Finding word neighbours with PageRank

Give hate speech words a higher importance.

Generate initial output for all words in HateWords as list



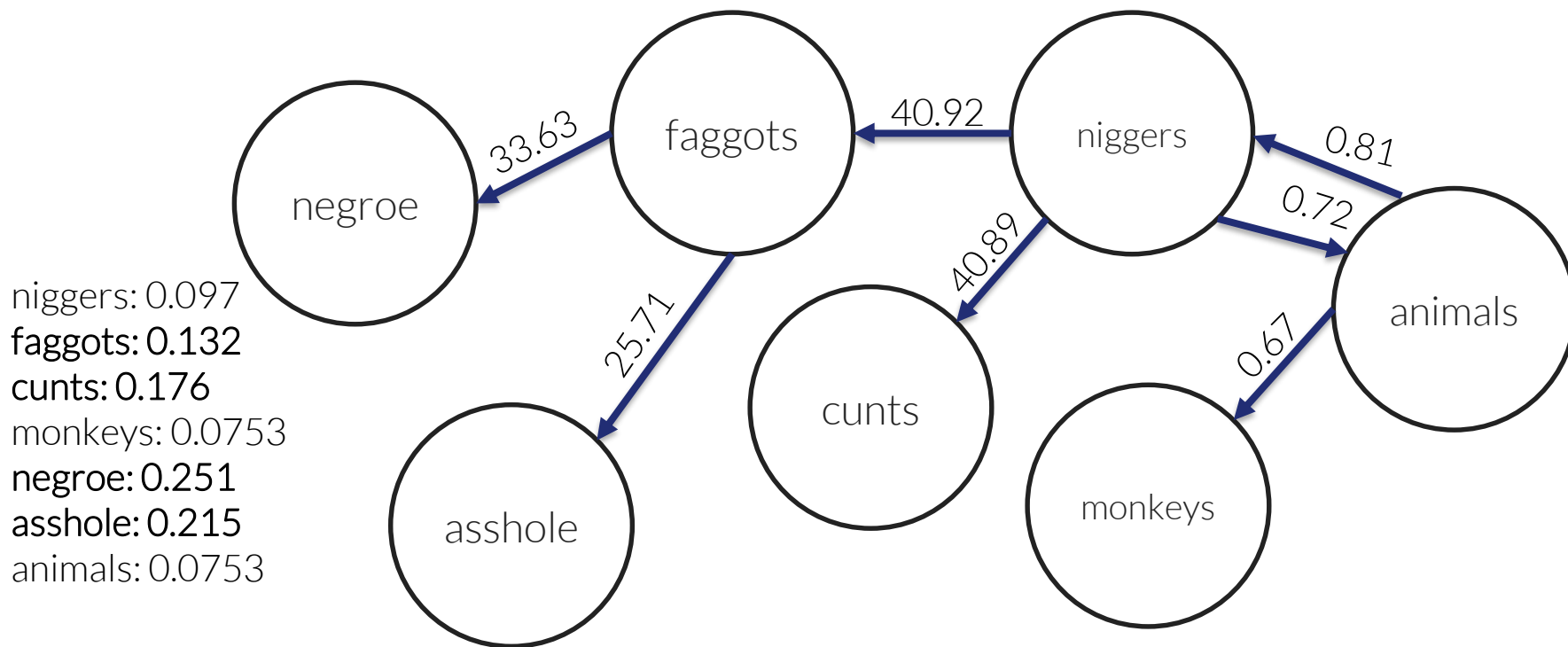
$count(nigger) = 10000$
 $boost(nigger) = 10$
 $cosineSim(nigger, faggots) = 0.92$

$count(animals) = 2000$
 $boost(animals) = 0$
 $cosineSim(animals, monkeys) = 0.67$

$wt(nigger, faggots) = \log(10000) * 10 + 0.92 = 40.92$
 $wt(animals, monkeys) = 0.67$

$$wt(v_1 v_2) = \begin{cases} \log(freq(v_1)) * boost(v_1) + cosineSim(v_1, v_2) & \text{if } v_1 \in boost \\ cosineSim(v_1, v_2) & \text{if } v_1 \notin boost \end{cases}$$

Finding word neighbours with PageRank



By doing this, known hate speech words pass on their weight

Allows us to find important neighbours of hate speech words

Trimming word neighbours

The graph can get huge, so we need to trim.

Trim the list

$$df = \frac{\text{doc_count}(w)}{N}$$

where w is a given word and N is number of documents in a dataset

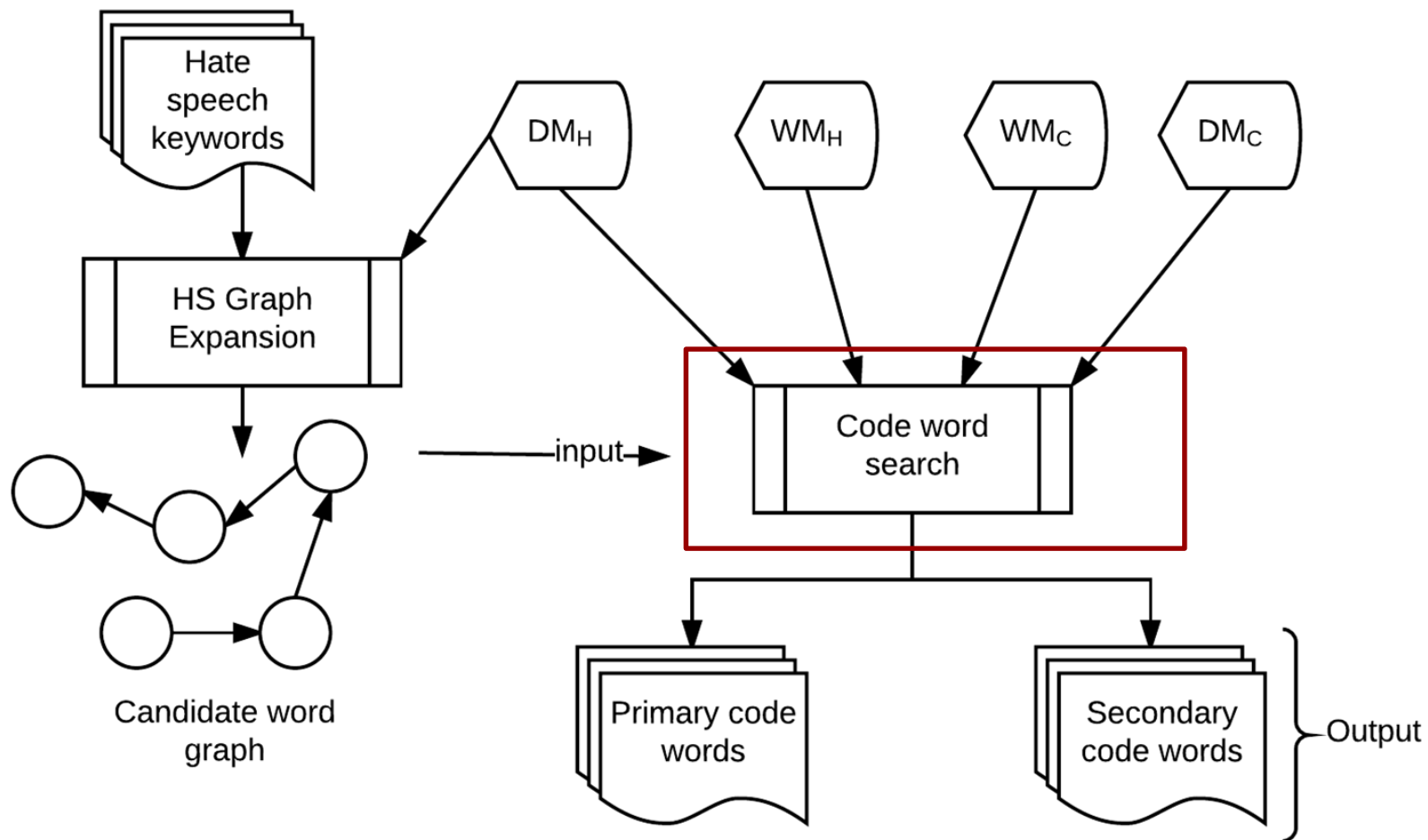
keep(w) if it appears in a higher fraction of HateCommunity than CleanTexts

remove(w) otherwise

remove(w) if is a known hate speech keywords

Unlikely that hate speech users would frequently use animals with its normal meaning

Code word ranking



Code word ranking

Code words are supposed to be secret.

Fetching **related** or **similar** words from **CleanTexts** is not enough to find hate speech code words.

Doing the same for **HateCommunity** might reveal a hate speech link but infer nothing about frequency of use.

We must also consider the frequency of a word in **HateCommunity** vs **CleanTexts**.

Code word ranking

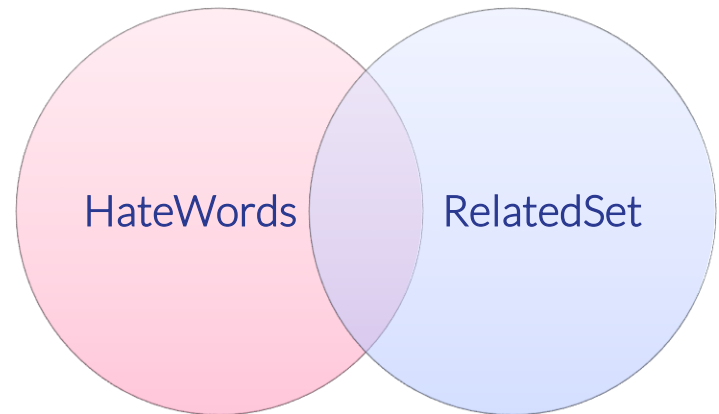
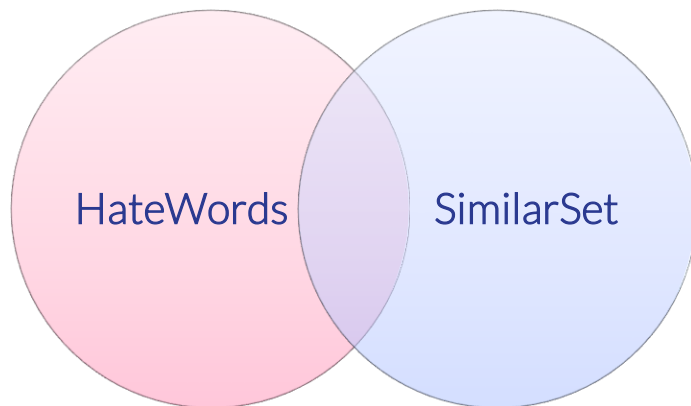
match@k = number of matches in top *k* word output

df = document frequency in a dataset

similar_set = similarity neighbours of a word *w* from HateCommunity

related_set = relatedness neighbours a word *w* from HateCommunity

frequency_{check} = $df(w \in HateTexts) > df(w \in CleanTexts)$



A word is a primary code word if:

$$\left(\frac{\text{size}(\text{either intersection})}{k} \right) \geq \text{match@k} \wedge \text{frequency}_{\text{check}}$$

Code word ranking: Secondary

We implement a secondary condition for words that fail Primary

With a word \mathbf{w} build a similarity graph \mathbf{G} as previously described

A word w is a secondary code word if for any $\mathbf{v} \in \mathbf{G}$

$$\mathbf{v} \in HW$$



Evaluation

Data Collection

Twitter data was collected during the

2016 US Presidential Elections

2017 US Presidential Inauguration

2017 Manchester bombing

Various points in 2017

CleanTexts and HateTexts: 10M tweets

HateCommunity: 400K tweets and articles

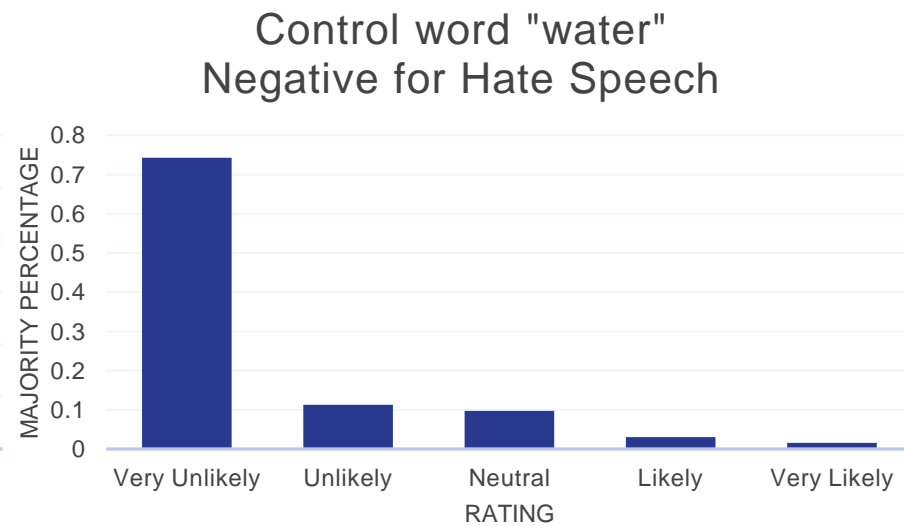
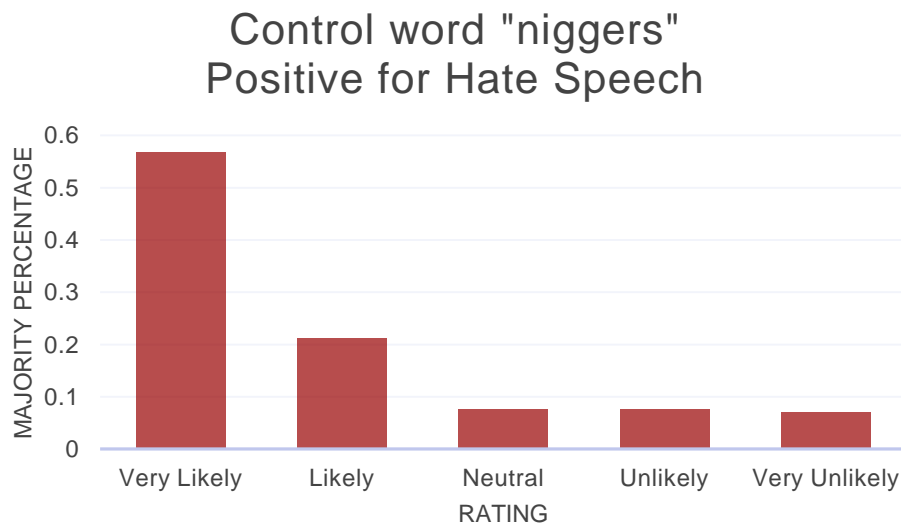
Evaluation Setup

55 Primary code words

262 Secondary code words

| | |
|----------------------------|---|
| Code words | Designed an experiment to see if annotators could infer the use |
| niggers [positive control] | of hate speech with only code words |
| snake | 5 samples for each word |
| googles | Ratings from Very unlikely to very likely [0 to 4] |
| cuckservatives | 3 separate experiments from different partitions |
| skypes | > 45 annotators for each experiment |
| creatures | |
| moslems | |
| cockroaches | |
| water [negative control] | <i>Another cop killed and set on fire by googles</i> |
| primitives | <i>Strange mixed breed creatures jailed for killing white woman</i> |
| | <i>Germany must disinfect her land. One cockroach at a time</i> |

Evaluation: Control results



Most participants were able to correctly answer the control across all 3 experiments

Evaluation: Annotator Agreement

Created ground truth and aggregate majority annotator ratings

Ratings > 2 were labelled as Hate Speech

Ratings < 2 were labelled as Not Hate Speech

Krippendorff's Alpha agreement scores using majority ratings

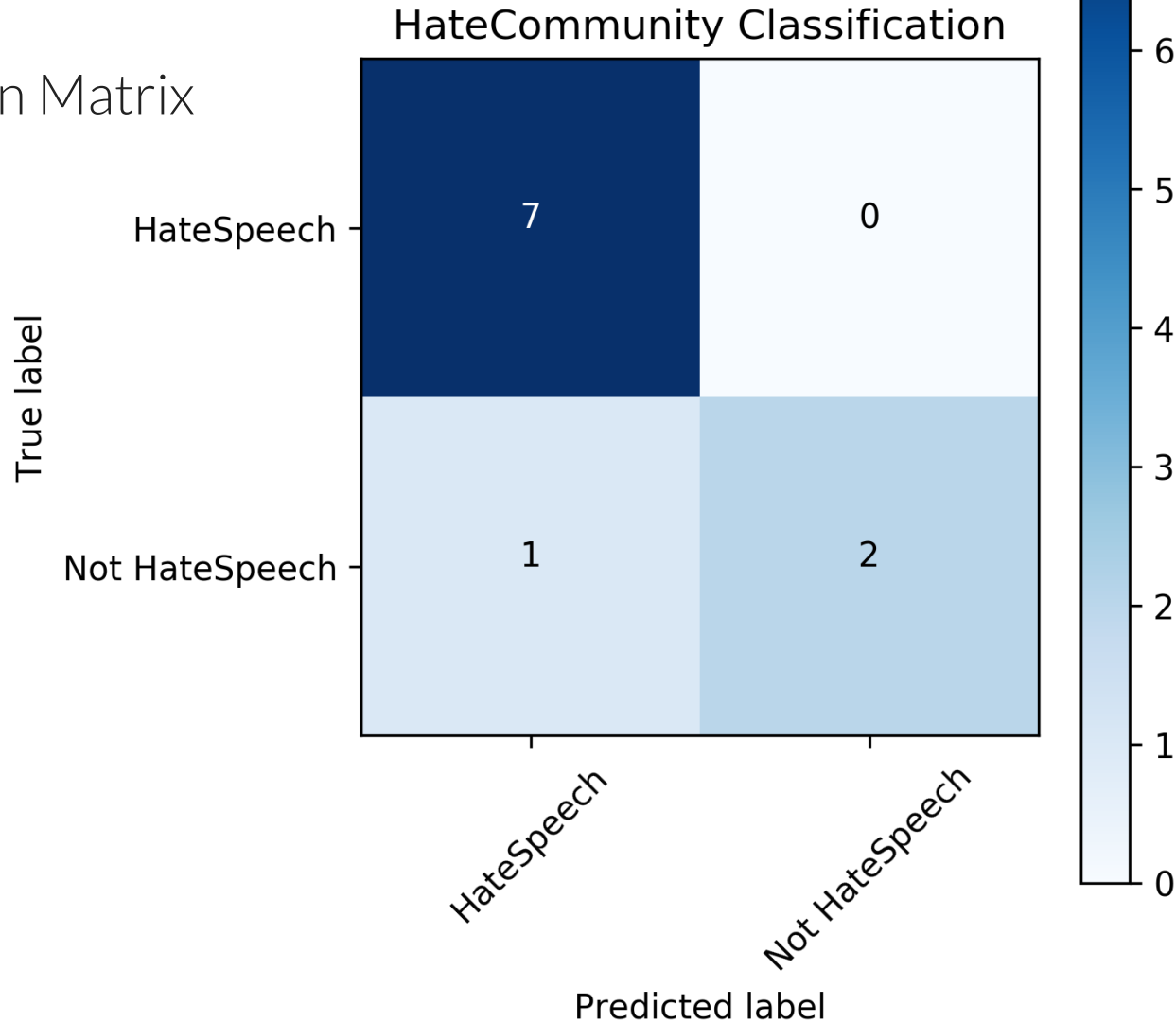
HateCommunity: $K=0.871$

CleanTexts: $K = 0.676$

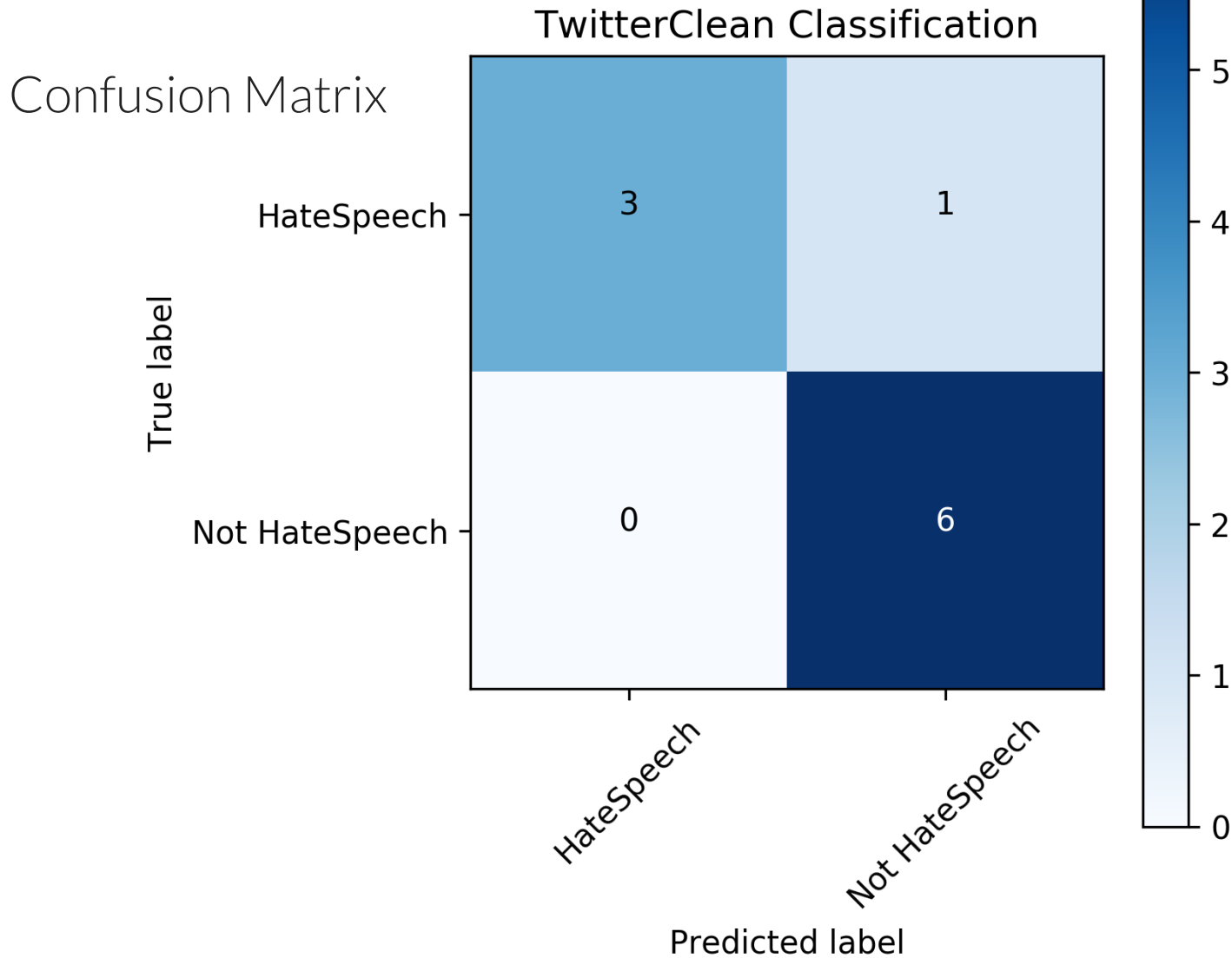
HateTexts: $K = 0.807$

Evaluation: Aggregate Classification

Confusion Matrix

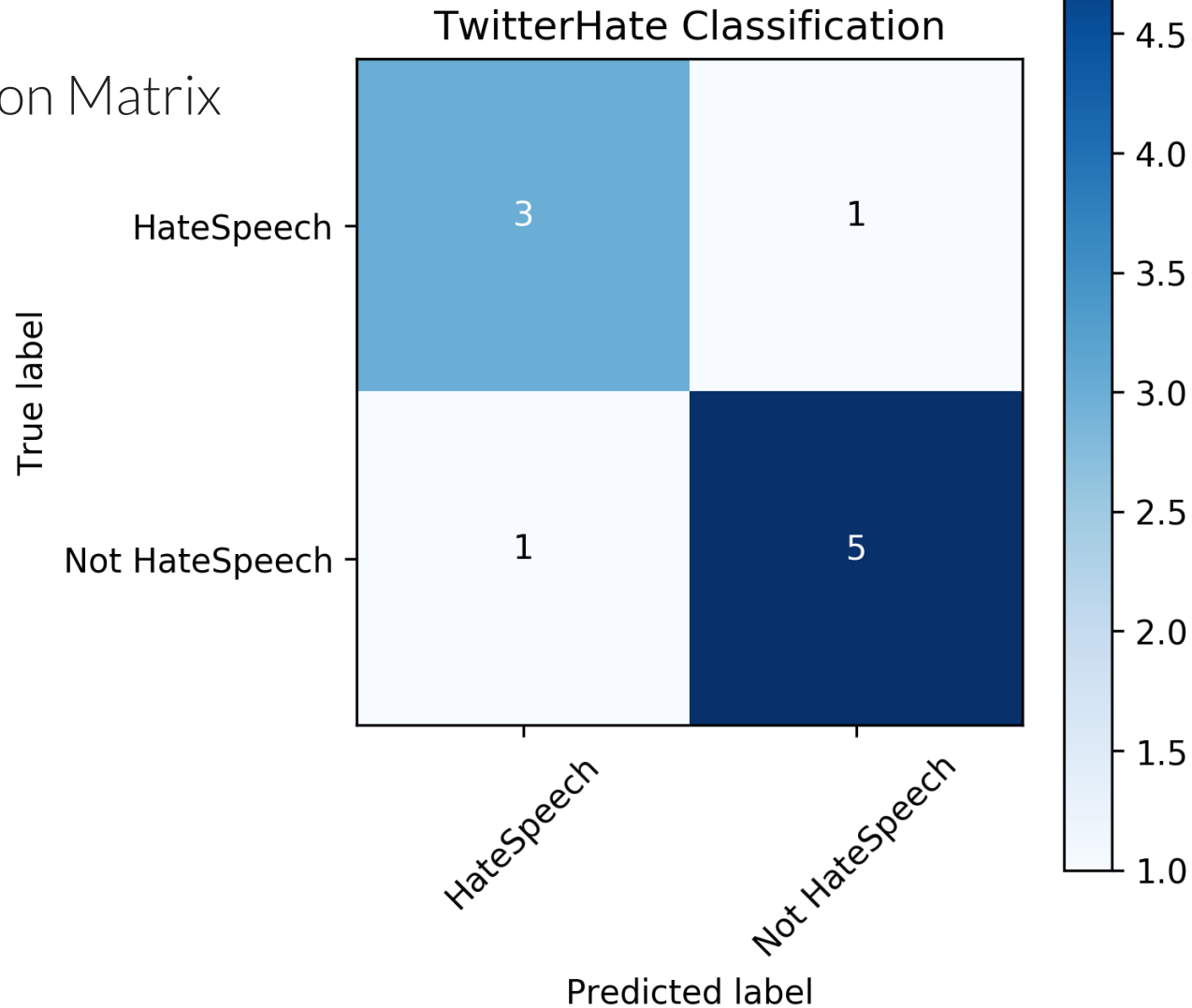


Evaluation: Aggregate Classification



Evaluation: Aggregate Classification

Confusion Matrix



Evaluation: Aggregate Classification

| | | Hate Speech | Not Hate Speech |
|---------------|-----------|-------------|-----------------|
| HateCommunity | Precision | 0.88 | 1.00 |
| | Recall | 1.00 | 0.67 |
| | F1 | 0.93 | 0.80 |
| CleanTexts | Precision | 1.00 | 0.86 |
| | Recall | 0.75 | 1.00 |
| | F1 | 0.86 | 0.92 |
| HateTexts | Precision | 0.75 | 0.83 |
| | Recall | 0.75 | 0.83 |
| | F1 | 0.75 | 0.83 |


Conclusions

We propose our dynamic method for learning out-of-dictionary hate speech code words.

Our annotation experiment showed that it is possible to infer hate speech without knowing the meaning of the code word.

Hate speech is not dependent on a vocabulary, it is an issue of context.

We hope to implement a method to automatically update our model.



Thank you.
Q & A.

Evaluation: Ranking Distribution

HateCommunity Results

| Words | Ground Truth | | Annotators | |
|----------------|---------------|---------------------|---------------|---------------------|
| | Label | Majority percentage | Label | Majority percentage |
| Niggers | Very likely | 0.8 | Very likely | 0.68 |
| Snakes | Unlikely | 0.4 | Neutral | 0.26 |
| Googles | Very likely | 1.0 | Very likely | 0.41 |
| Cuckservatives | Unlikely | 0.8 | Likely | 0.36 |
| Skypes | Likely | 0.8 | Likely | 0.3 |
| Creatures | Very likely | 0.6 | Very likely | 0.4 |
| Moslems | Likely | 0.8 | Very likely | 0.39 |
| Cockroaches | Very likely | 1.0 | Very likely | 0.40 |
| Water | Very unlikely | 1.0 | Very unlikely | 0.65 |
| Primitives | Very likely | 0.6 | Very likely | 0.37 |